# Quartz 2D in Depth

Session 207

# Quartz 2D in Depth

**Derek Clegg**
Quartz Engineering

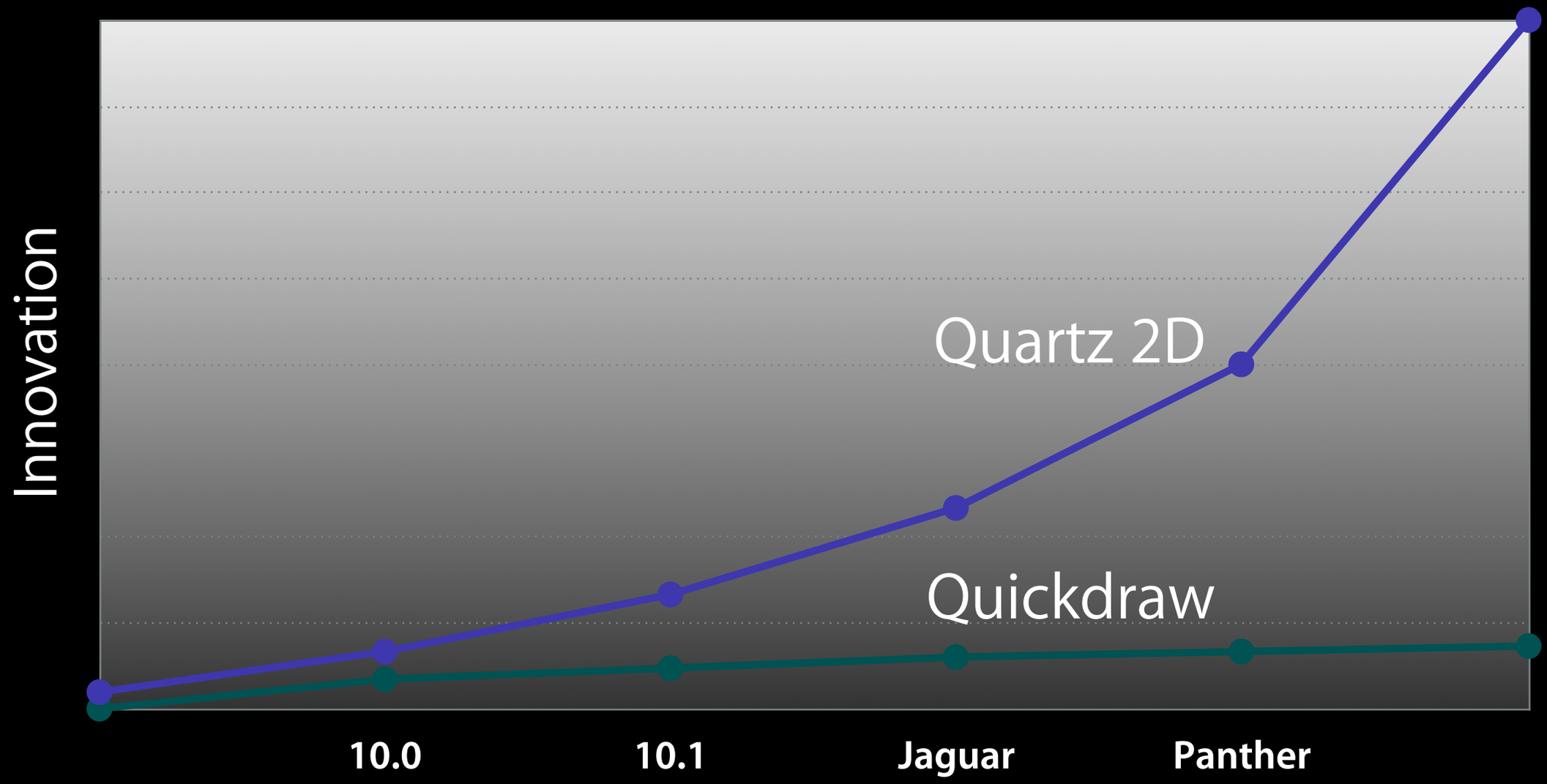
# Agenda

- What is Quartz?
- Quartz Architecture
- Quartz API
- Demos

# What Is Quartz?

- Quartz is the Mac OS X imaging model
- Influenced by industry standard graphics systems
  - PostScript, PDF, QuickDraw, Java2D,…
- The next-generation imaging model for the next-generation applications

# Where We're Going



# Quartz 2D

- Low-level, lightweight 2D rendering library
- Resolution and device independent
- Uses Apple Type Services for font management
- Uses ColorSync for color management

# Who Uses Quartz 2D?

- Carbon, Cocoa, Java in Mac OS X
- Most Apple applications
  - Safari, iApps, Keynote,…
- Third-party applications
  - Mathematica, Office, DeltaGraph,…
- Your application

# Quartz Compositor

- Handles all window management
- Handles compositing of windows onto display
- Learn more in session 211, “Introduction to Quartz Services”

# Quartz 2D Features

- High-quality 2D graphics
  - Anti-aliased rendering
  - Transparency
- Offscreen rendering
- PDF document access, rendering, and creation

# CoreGraphics Types

Contexts

Paths

Fonts

Images

PDF documents

Colors/Color Spaces

Shadings

Patterns

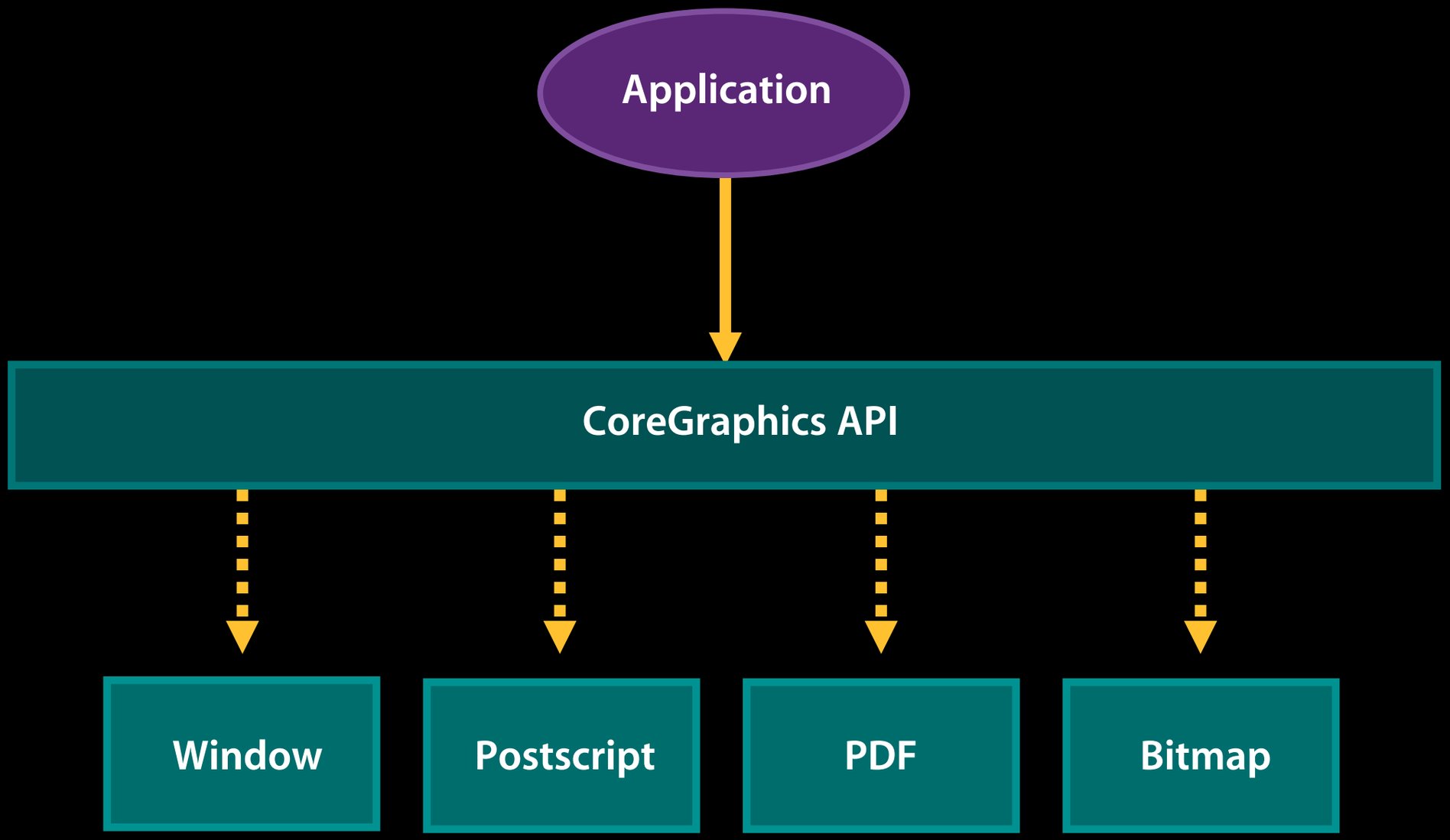
Geometry/Affine transforms

Data “managers”

# Quartz 2D In-Depth

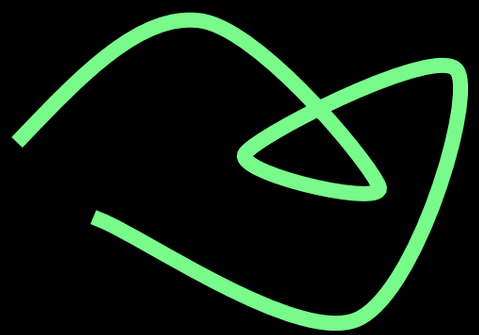
- Last year's WWDC session covers many "basics"—see the DVD for details

# CGContexts


Application
CoreGraphics API
Window
Postscript
PDF
Bitmap

# Drawing Primitives

Paths:



Text:

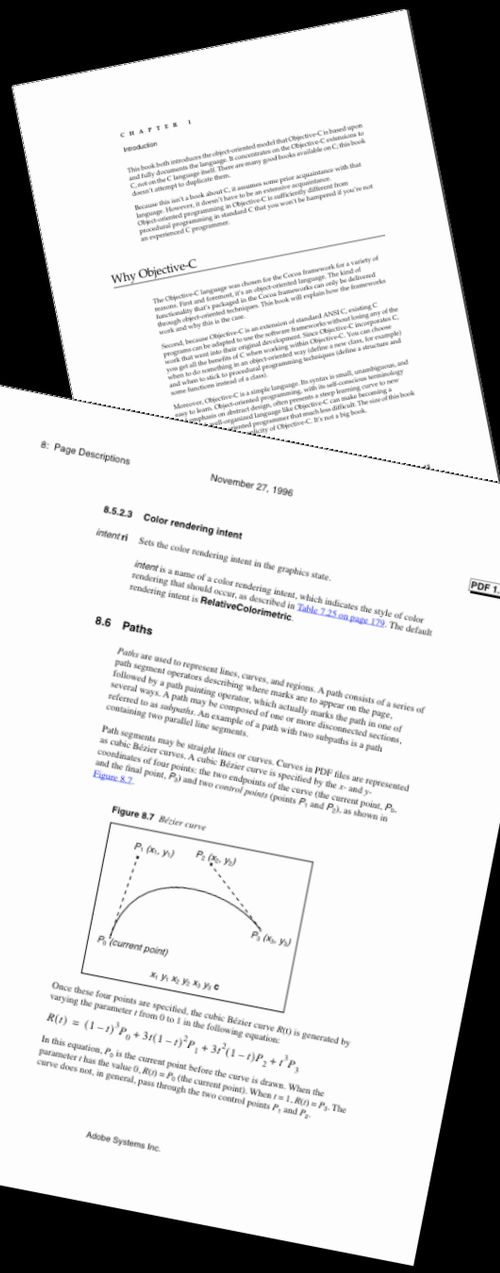
Welcome

Rotation
Rotation
Rotation
Rotation
Rotation

Images:

# Drawing Primitives

PDF Pages:



CHAPTER 1

Introduction

This book both introduces the object-oriented model that Objective-C is based upon and fully documents the language. It concentrates on the Objective-C extensions to C, not on the C language itself. There are many good books available on C; this book doesn't attempt to duplicate them.

Because this isn't a book about C, it assumes some prior acquaintance with that language. However, it doesn't have to be an extensive acquaintance. Object-oriented programming in Objective-C is sufficiently different from procedural programming in standard C that you won't be hampered if you're not an experienced C programmer.

## Why Objective-C

The Objective-C language was chosen for the Cocoa framework for a variety of reasons. First and foremost, it's an object-oriented language. The kind of functionality that's packaged in the Cocoa frameworks can only be delivered through object-oriented techniques. This book will explain how the frameworks work and why this is the case.

Second, because Objective-C is an extension of standard ANSI C, existing C programs can be adapted to use the software frameworks without losing any of the work that went into their original development. Since Objective-C incorporates C, you get all the benefits of C when working within Objective-C. You can choose when to do something in an object-oriented way (define a new class, for example) and when to stick to procedural programming techniques (define a structure and some functions instead of a class).

Moreover, Objective-C is a simple language. Its syntax is small, unambiguous, and easy to learn. Object-oriented programming, with its self-conscious terminology and emphasis on abstract design, often presents a steep learning curve to new object-oriented programmer that much less difficult. The size of this book is a testament to the simplicity of Objective-C. It's not a big book.

8: Page Descriptions

November 27, 1996

**8.5.2.3 Color rendering intent**

*intent* **ri** Sets the color rendering intent in the graphics state.

*intent* is a name of a color rendering intent, which indicates the style of color rendering that should occur, as described in Table 7.25 on page 179. The default rendering intent is **RelativeColorimetric**.

PDF 1.1

## 8.6 Paths

*Paths* are used to represent lines, curves, and regions. A path consists of a series of path segment operators describing where marks are to appear on the page, followed by a path painting operator, which actually marks the path in one of several ways. A path may be composed of one or more disconnected sections, referred to as *subpaths*. An example of a path with two subpaths is a path containing two parallel line segments.

Path segments may be straight lines or curves. Curves in PDF files are represented as cubic Bézier curves. A cubic Bézier curve is specified by the *x*- and *y*-coordinates of four points: the two endpoints of the curve (the current point, $P_0$, and the final point, $P_3$) and two *control points* (points $P_1$ and $P_2$), as shown in Figure 8.7.

**Figure 8.7** *Bézier curve*



Once these four points are specified, the cubic Bézier curve $R(t)$ is generated by varying the parameter $t$ from 0 to 1 in the following equation:

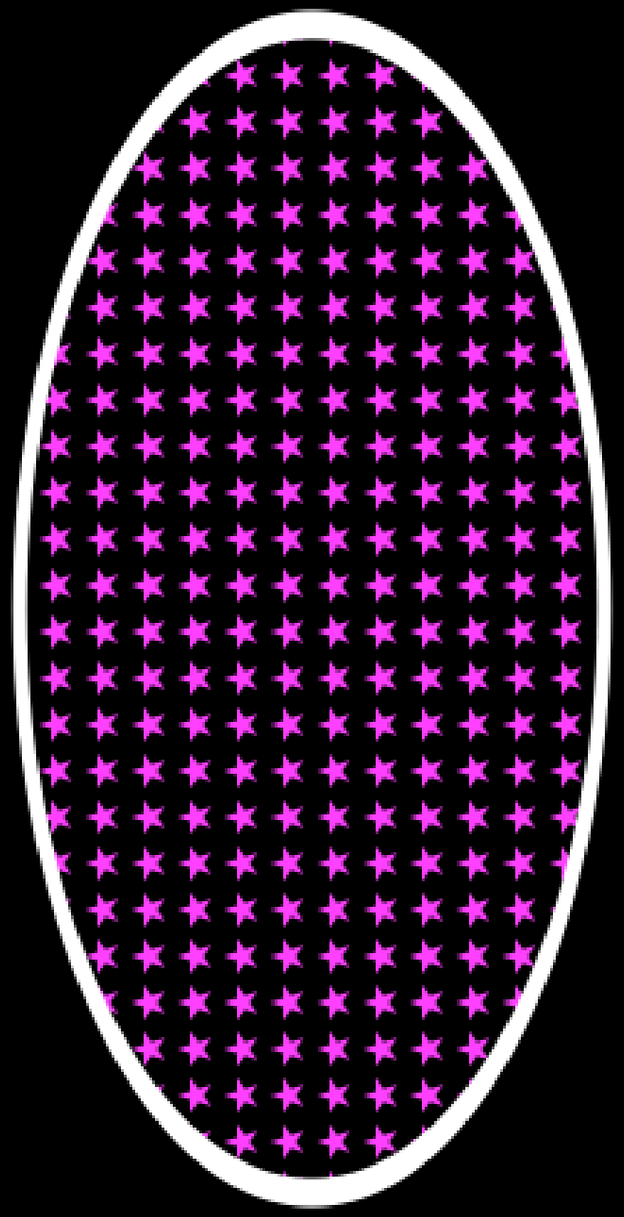
$$R(t) = (1-t)^3 P_0 + 3t(1-t)^2 P_1 + 3t^2(1-t) P_2 + t^3 P_3$$

In this equation, $P_0$ is the current point before the curve is drawn. When the parameter $t$ has the value 0, $R(t) = P_0$ (the current point). When $t = 1$, $R(t) = P_3$. The curve does not, in general, pass through the two control points $P_1$ and $P_2$.

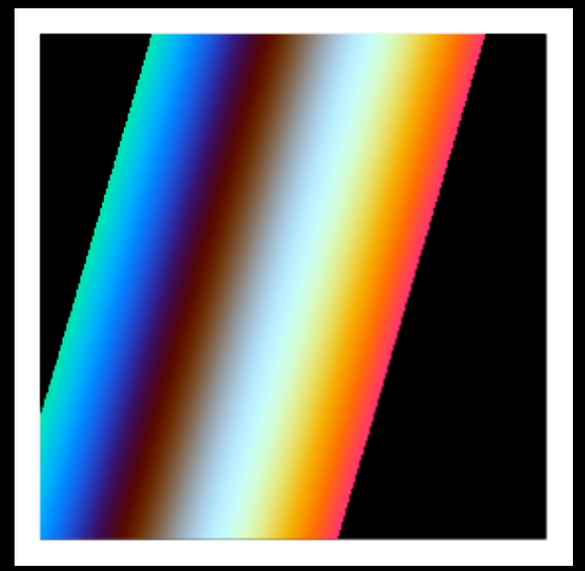
222

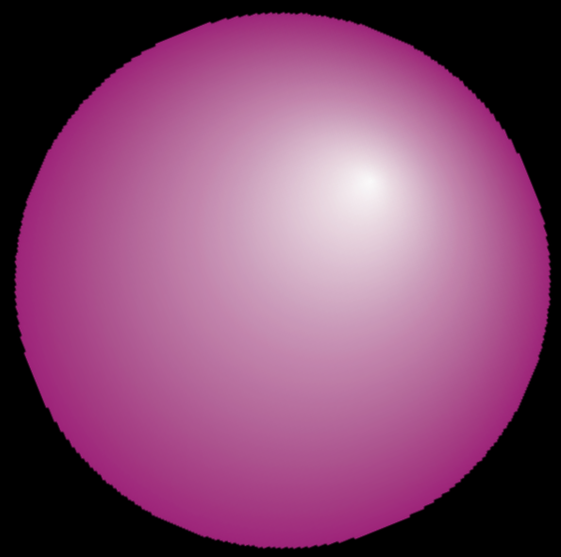
Adobe Systems Inc.

Patterns:



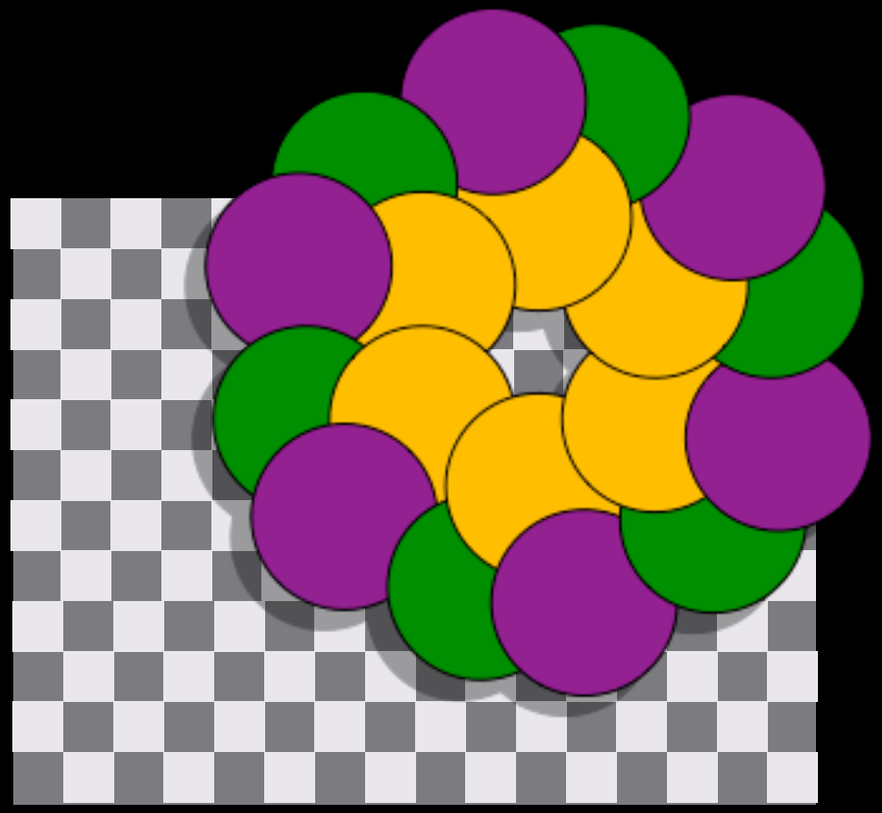
Shadings:



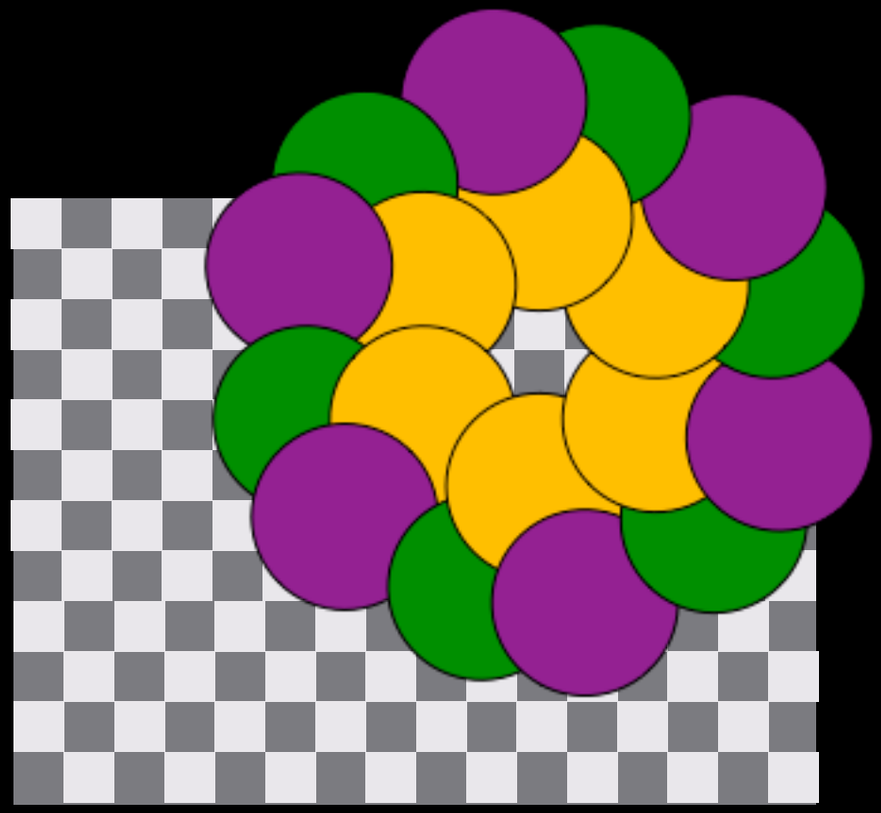


# Drawing Primitives

new Shadows:



new Transparency Layers:



# Vector Geometry

# Vector Geometry Clipping

# Text

Welcome 겊경과굿

瑚うり桓ずゐき

Stroked

Rotated

# Glyph Rendering

- CG works with glyphs, not “text”
- Faster than Jaguar—up to 2 times as fast
- Font support:  TrueType, Type1, OpenType, CID through ATS
- Use ATSUI or Cocoa for Unicode and layout support

# Images

# Images

- Alpha channel (for transparency)
- 1- to 32-bits per component
- Color space support
  - RGB, CMYK, Indexed, ICC profiles via ColorSync
- 1- to 8-bit image masks

# Images With Profiles

RGB profile

BGR profile

# QuickTime and Quartz

- QuickTime API to create a CGImage from a graphics importer
- Allows any QuickTime-supported image type to be drawn into a CGContext
  - GIF, TIFF, JPEG2000, …
- Preserves as much information as possible

# QuickTime and Quartz

```
GraphicsImportCreateCGImage(
    GraphicsImportComponent ci,
    CGImageRef *imageRefOut,
    UInt32 flags);
```

# PDF Documents

- Import PDF 1.0, 1.1, … 1.4 documents
- Export PDF 1.3 documents
  - PDF 1.4 if required
- Full round-tripping of CG API
- Prepress data interchange via PDF/X-3: 2002

# new PDF Page APIs

- Page-based API for PDF information
  - Media box, crop box, art box
  - Rotation angle
  - Page number

# new PDF Metadata Access

- New types and API for PDF data
  - `CGPDFDictionary, CGPDFArray, ...`
- New API to get a document's "catalog"
  - `CGPDFDocumentGetCatalog(...)`
- New API to get a page's dictionary
  - `CGPDFPageGetDictionary(...)`

# PDF Metadata Example

```
CGPDFDictionaryRef d;
CGPDFStreamRef stream;
d = CGPDFPageGetDictionary(page);
if (CGPDFDictionaryGetStream(d, "Thumb", &stream)) {
    data = CGPDFStreamCopyData(stream, &format);
    ...
```

# Demo

# Color Spaces

- CGColorSpace contains color-interpretation data
- new User's "document-default" color space
- new Display color space:  fast (no match) when drawing to screen; "generic" profile when printing
- See Session 206, "ColorSync Update," for more info

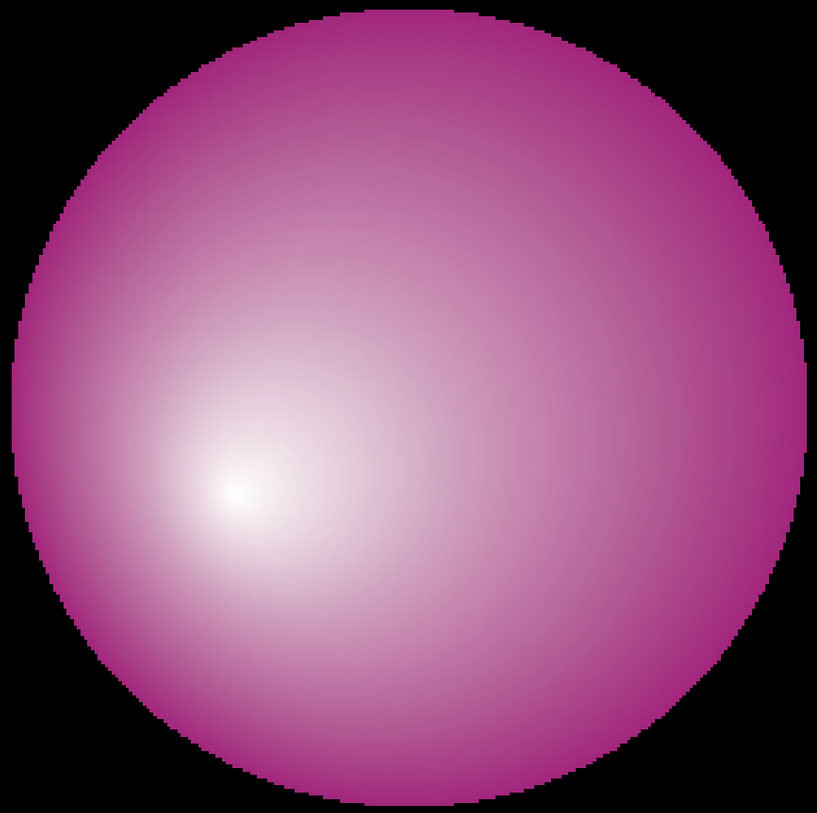
# Colors

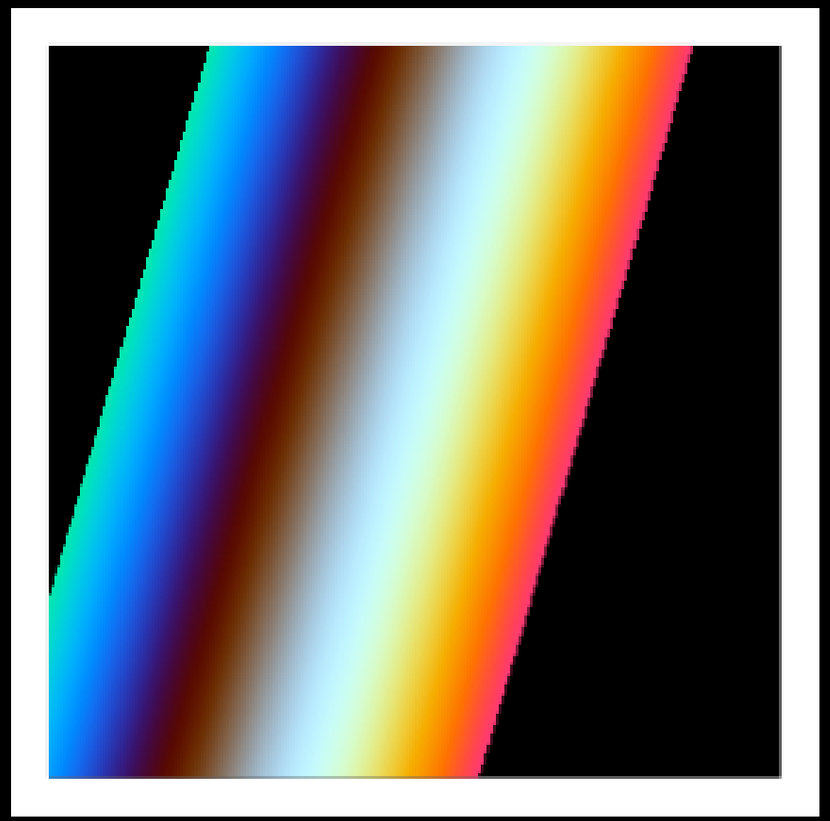
- New type: `CGColorRef`
- Container for color space and components
- Fundamental type for colors in Quartz 2D

# Colors

- Fast!
- Most useful with “constant” colors
- New API for setting colors directly:
  - `CGContextSetFillColorWithColor`
  - `CGContextSetStrokeColorWithColor`

# Shadings

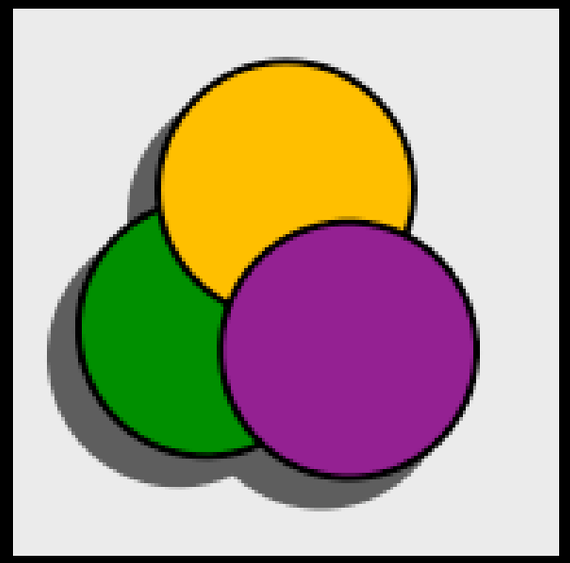


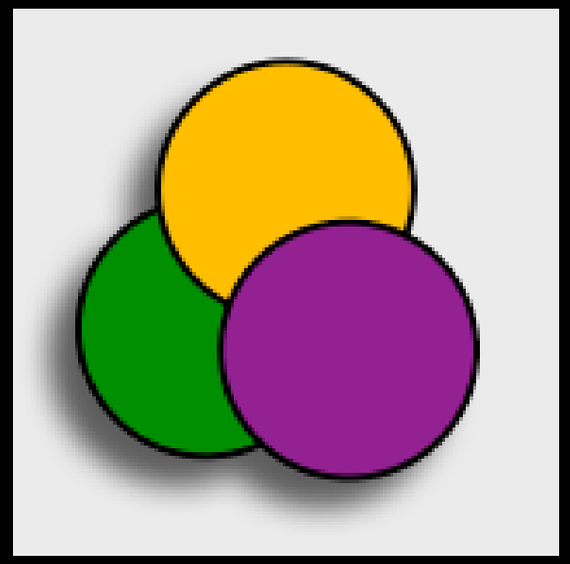


# Shadings

- Better performance for axial and radial shadings
- Better rendering—controlled by interpolation quality
- new Dithering
- new Shadings work with non-opaque colors

# Shadows





# Shadows

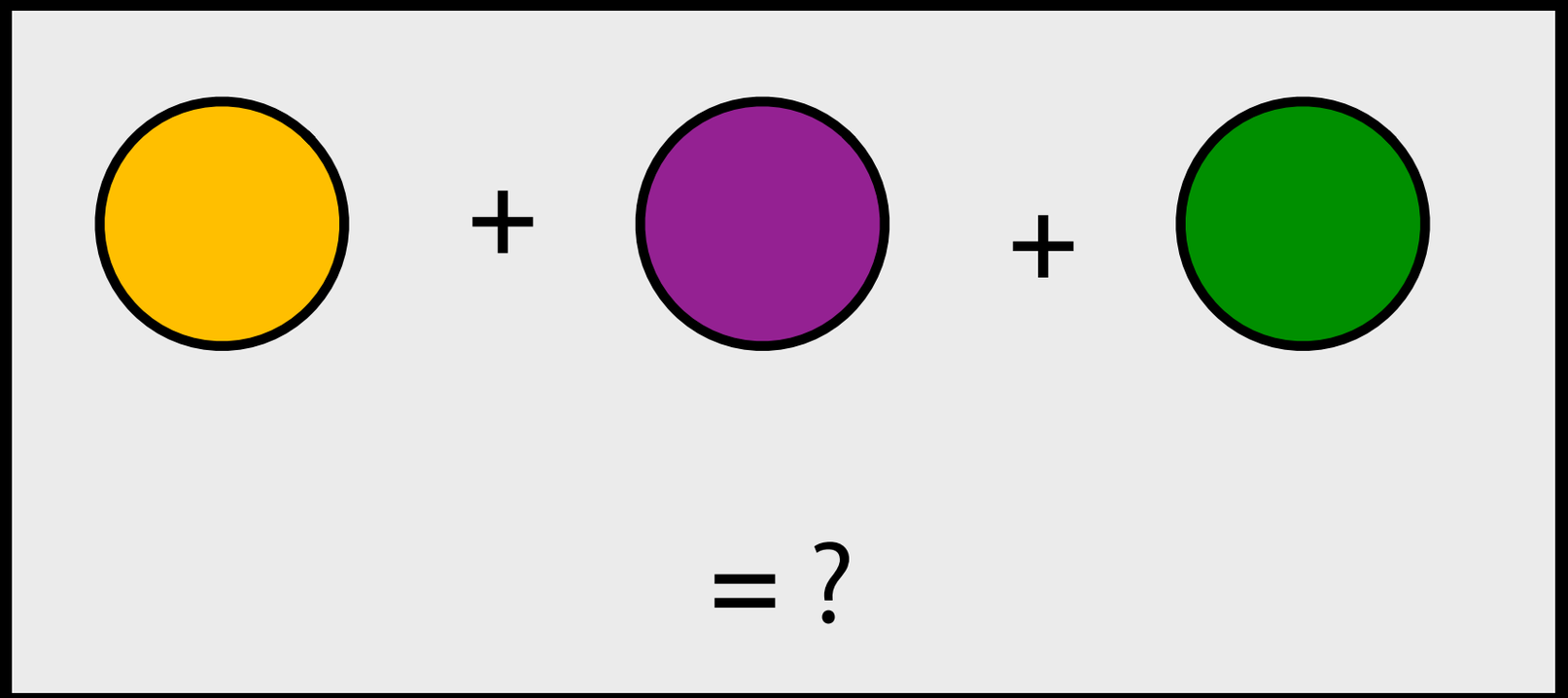
- API specifies offset and blur radius
- Part of the graphics state

# Shadow Example

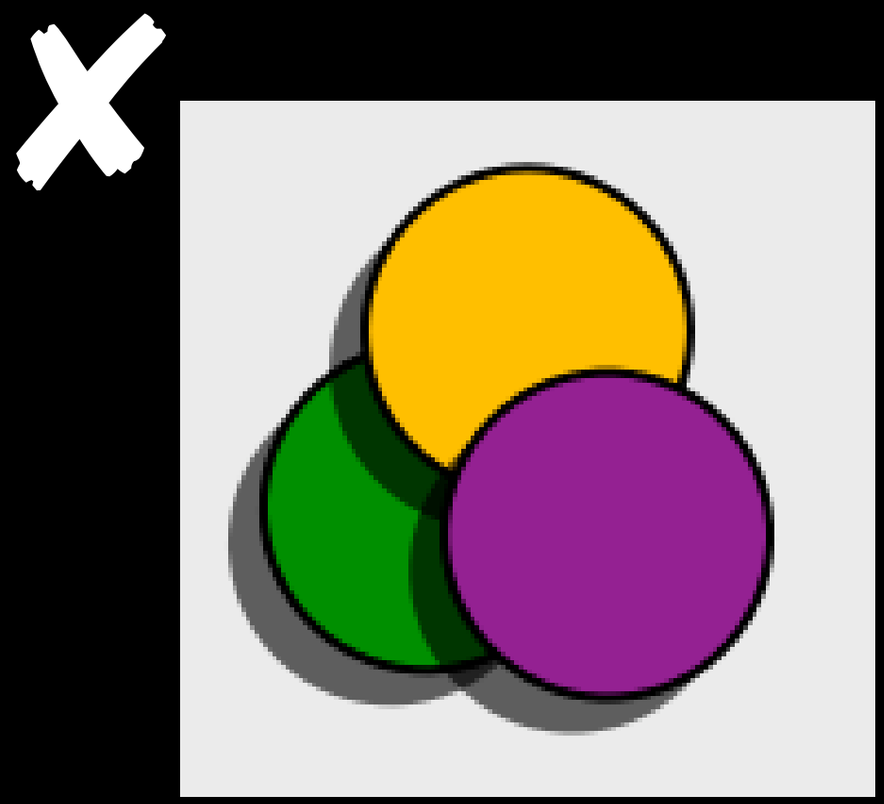
```
/* Offset specified in default user space. */
CGContextSaveGState(context);
CGSize offset = CGSizeMake(5, 5);
CGContextSetShadow(context, offset, blurRadius);
CGContextStrokePath(context);
CGContextRestoreGState(context);
```

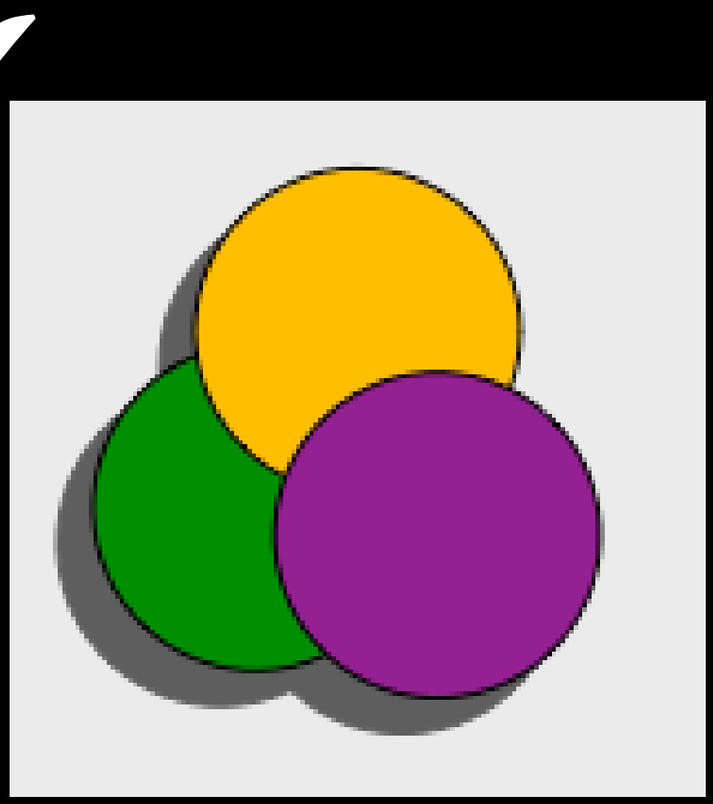
# Demo

# Transparency Layers



# Transparency Layers





# Transparency Layers

- Useful for rendering groups of objects and applying an effect to the result
- Similar to a layer in graphics applications
- Graphics state parameters unchanged except alpha, shadow, and other parameters affecting final composite
- Nestable!

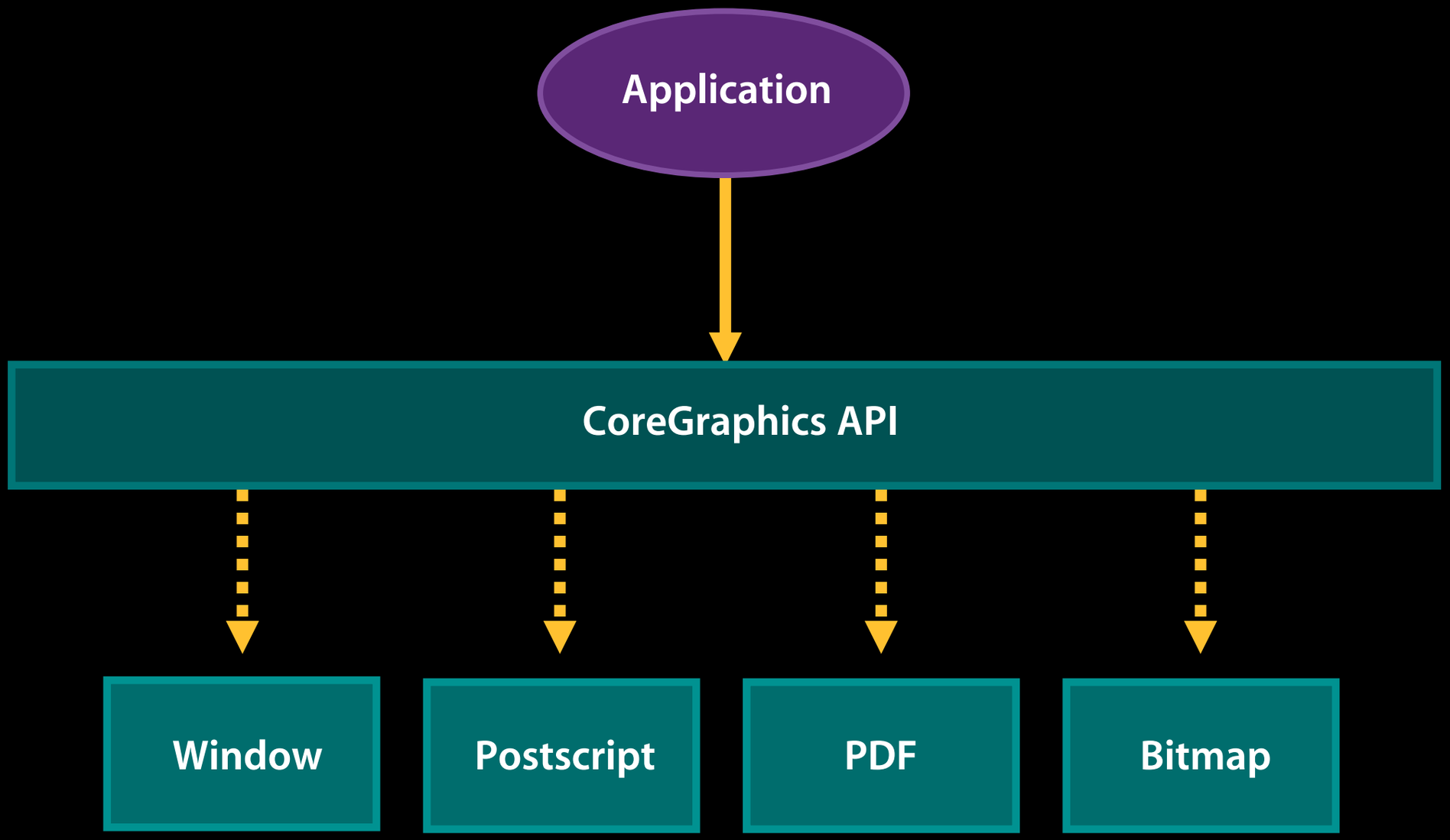
# Transparency Layers

# Transparency Layers Example

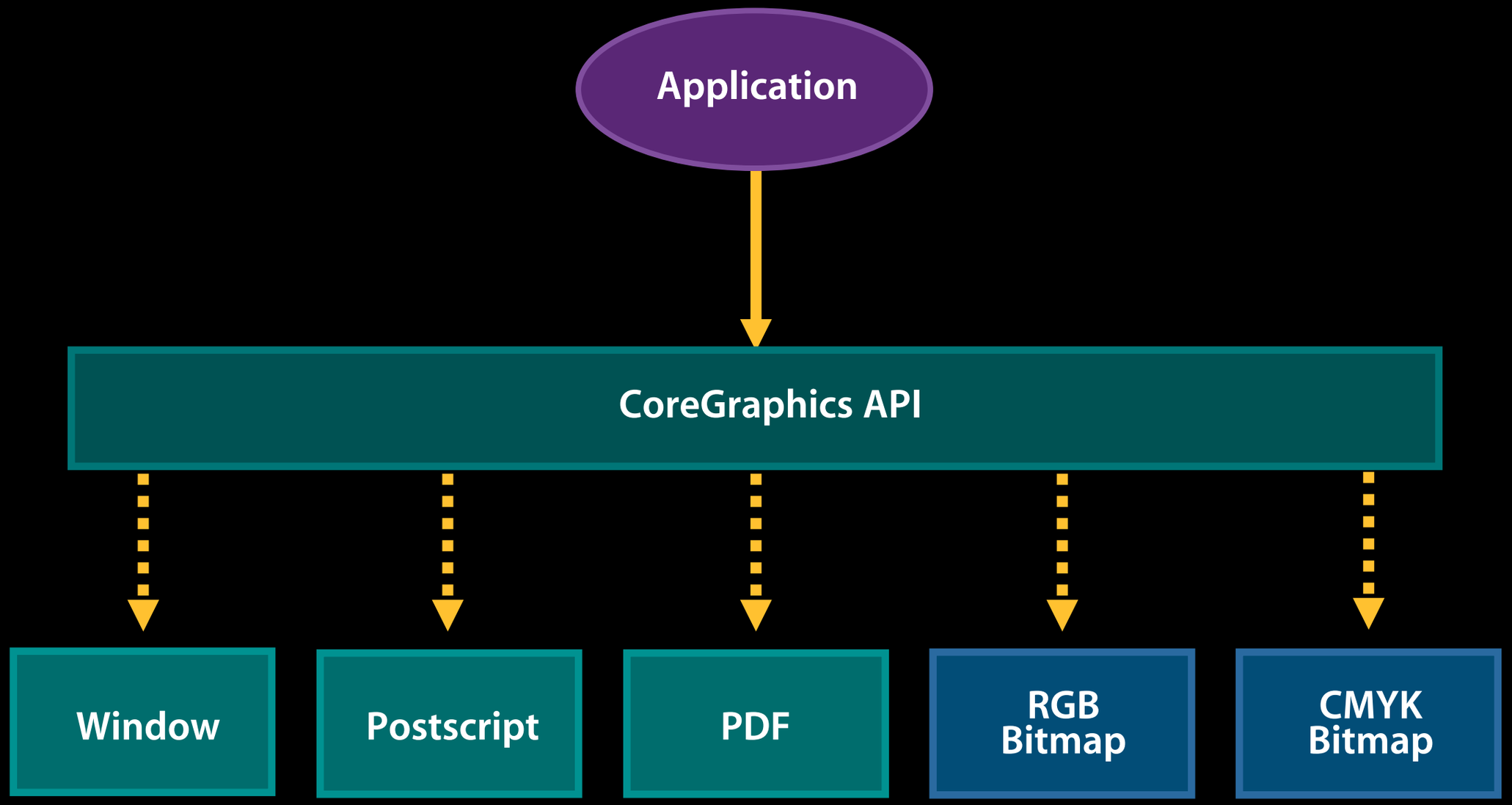
```
/* Set the alpha for the final composite. */
CGContextSetAlpha(context, 0.7);
CGContextBeginTransparencyLayer(context, NULL);
drawCirclesInRect(context);
/* Composite the result of drawing into the context. */
CGContextEndTransparencyLayer(context);
```

# Demo

# CGContexts


Application
CoreGraphics API
Window
Postscript
PDF
Bitmap

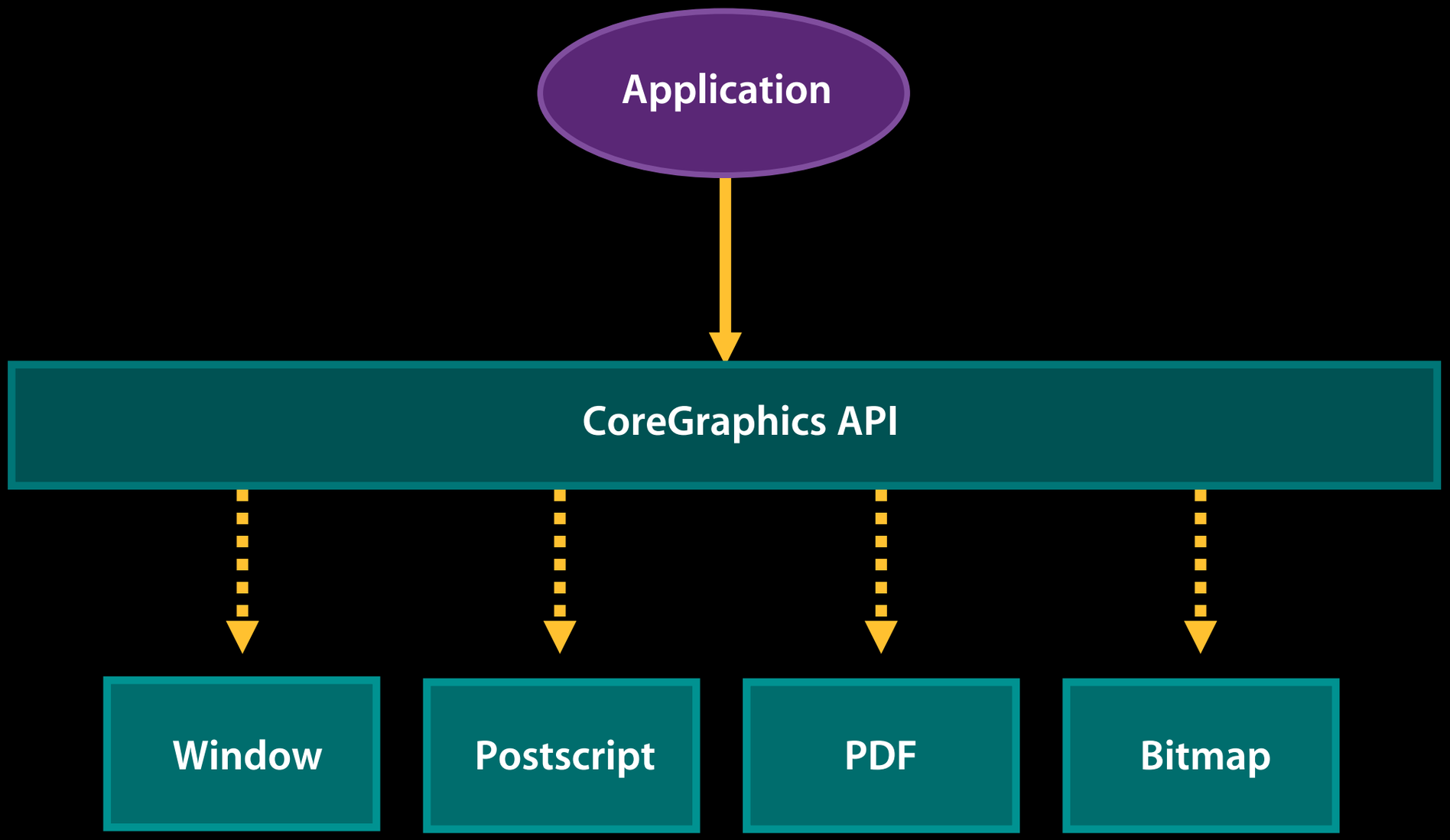
# CGContexts


Application
CoreGraphics API
Window
Postscript
PDF
RGB Bitmap
CMYK Bitmap

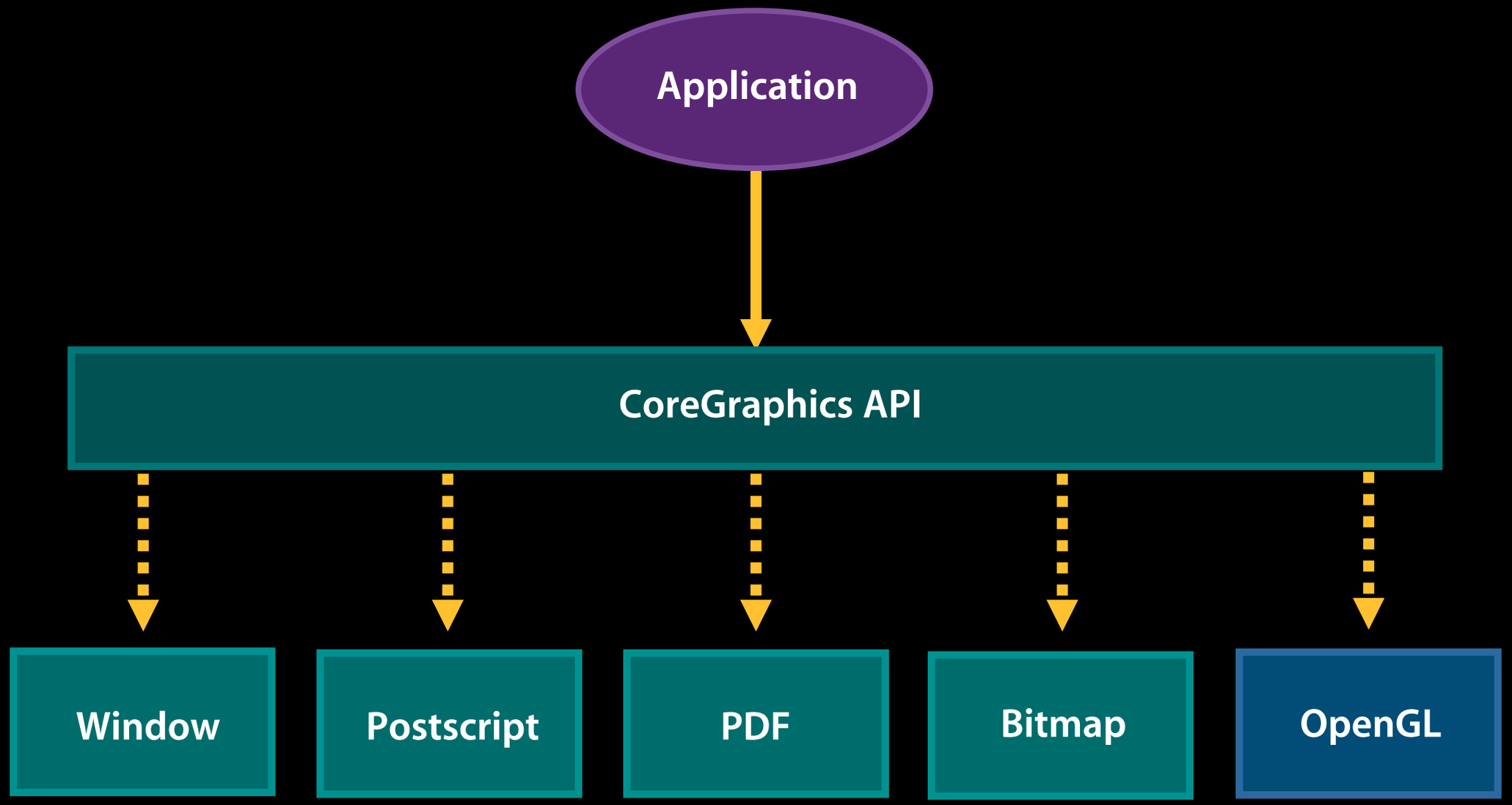
# CMYK Bitmap Context

- No new API—just use a CMYK colorspace when you create the context

# CGContexts


Application
CoreGraphics API
Window
Postscript
PDF
Bitmap

# CGContexts


Application
CoreGraphics API
Window
Postscript
PDF
Bitmap
OpenGL

# new OpenGL Context

- New API to create an OpenGL context from a CGLContextObj

```
CGContextRef CGGLContextCreate(
        CGLContextObj glContext,
        CGSize size,
        CGColorSpaceRef colorspace);

void CGGLContextUpdateViewportSize(
        CGContextRef c, CGSize size);
```

# new Pixel Copying API

- ScrollRect and CopyBits “lite”
- “Scroll”—that is, copy—bits in a rectangle to another rectangle
  - Source and destination may be the same or different contexts
- Create a CGImage from a rectangle in a context
- Not in this seed, but soon…

# Context Summary

- Many new context APIs
- PDF meta-data access
- Pixel copying
- Performance/quality improvements

# Python Bindings

- Easy access to the powerful Quartz rendering/ document model
- Interpreted object-oriented language affords rapid “one-off” development
- Many possible uses
  - Scripting of PDF manipulation
  - Bitmap rendering

# Python Bindings

- Most APIs are wrapped
  - CGBitmapContext, CGColor, CGImage, and so on
- Extra convenience APIs
  - QuickTime image I/O
  - Layout/rendering of plain text, HTML, RTF

# Python Example

```
from CoreGraphics import *
pageRect = CGRectMake(0, 0, 612, 792)
ctx = CGPDFContextCreateWithFilename
        ("circle.pdf", pageRect)
ctx.beginPage(pageRect)
ctx.setRGBFillColor(1, 0, 0, 1)
ctx.addArc(300, 300, 100, 0, 2*math.pi, 1)
ctx.fillPath()
ctx.endPage()
```

# PostScript to PDF

- New API to convert PostScript files to PDF
- Callback mechanism for per-page progress
- Caveats
  - Entire PostScript VM in your process
  - Can be slow to start up

# PostScript to PDF Example

```
in = CGDataProviderCreateWithURL(url);
out = CGDataConsumerCreateWithURL(url);
converter = CGPSConverterCreate(NULL,
        &callbacks, NULL);
success = CGPSConverterConvert(converter,
        in, out, NULL);
/* Could reuse converter at this point. */
```

# Summary

- Lots of new API to play with
- Continuing to add API where appropriate
- As always, your feedback is encouraged

# Wrap Up

# Roadmap

| | |
|---|---|
| **200 Graphics and Imaging Overview**<br>Latest developments in Mac OS X graphics APIs. | **Presidio**<br>**Tues., 10:30am** |
| **201 Mac OS X OpenGL in Depth**<br>Introduction to OpenGL on Mac OS X. | **Nob Hill**<br>**Tues., 2:00pm** |
| **202 Technology Magic of Keynote**<br>How Keynote uses Mac OS X graphics technologies. | **Nob Hill**<br>**Tues., 3:30pm** |
| **204 Image Capture Update**<br>Integrating Image Capture with your product. | **Marina**<br>**Wed., 9:00am** |

# Roadmap

| | |
|---|---|
| **205 Vertex Programming with OpenGL**<br>Super-fast geometry transformations using the GPU. | **Marina**<br>**Wed., 3:30pm** |
| **206 ColorSync Update**<br>Latest developments in Mac OS X color management. | **Mission**<br>**Wed., 5:00pm** |
| **207 Quartz 2D in Depth**<br>Harnessing the power of Quartz 2D in your product. | **Presidio**<br>**Thu., 9:00am** |
| **208 Fragment Programming with OpenGL**<br>Creating incredible per-pixel effects on Mac OS X. | **Pacific Heights**<br>**Thu., 10:30am** |

# Roadmap

| | |
|---|---|
| **209 OpenGL Optimizations**<br>Obtaining maximum 3D performance on Mac OS X. | **Marina**<br>**Thurs., 2:00pm** |
| **210 Mac OS X Printing Update**<br>Latest developments in the Mac OS X printing system. | **Mission**<br>**Thurs., 3:30pm** |
| **211 Introduction to Quartz Services**<br>Capturing & configuring displays on Mac OS X. | **Pacific Heights**<br>**Fri., 9:00am** |
| **212 Cutting-Edge OpenGL Techniques**<br>Expert recipes for stunning 3D effects. | **Russian Hill**<br>**Fri., 2:00pm** |

# Roadmap

| | |
|---|---|
| **421 Mac OS X Accessibility**<br>Latest developments in Mac OS X accessibility. | **Presidio**<br>**Fri., 10:30am** |
| **FF018 Graphics & Imaging Feedback**<br>Share your thoughts on graphics & imaging technologies. | **North Beach**<br>**Fri., 5:00pm** |

# Who to Contact


**Derek Clegg**
Quartz Technologies
dclegg@apple.com

**Travis Brown**
Graphics & Imaging Evangelist
travis@apple.com


http://developer.apple.com/wwdc2003/urls.html

# For More Information

# Reference Library

## Documentation

### Graphics & Imaging Documentation

ADC Home > Documentation > Graphics & Imaging >

### Quartz Developer Website

http://developer.apple.com/quartz/

# Q&A

**Travis W. Brown**
Graphics & Imaging Evangelist
travis@apple.com

http://developer.apple.com/wwdc2003/urls.html

# WWDC 2003

Worldwide Developers Conference